REMARKS

This application has been carefully considered in connection with the Examiner's Office Action dated July 26, 2007. Reconsideration and allowance are respectfully requested in view of the following.

Summary of Rejections

Claims 1 – 32 and 47-53 were pending at the time of the Office Action.

Claims 1 – 32 and 47 - 53 were rejected under 35 USC § 103(a) as being unpatentable over Miller et al, U.S. Patent No. 5,754,855 (hereinafter "Miller") in view of Obin et al, U.S. Patent No. 6,526,569 (hereinafter "Obin").

Claims 1-32 and 47-53 were rejected under 35 USC § 103(a) as being unpatentable over Miller in view of Obin and further in view of Keith Haviland, Dina Gray and Ben Salama, Pearson Education Limited, 1998 "Unix System Programming", second edition.

Summary of Response

Claims 1 and 5-10 are amended.

Claims 33-46 and 54 are canceled.

Claims 55-69 are new.

Claim 20 was previously presented.

Claims 2-4, 11-19, 21-32, and 47-53 remain as originally filed.

Summary of Claims Pending

Claims 1-32, 47-53, and 55-69 are currently pending following this response.

Applicant Initiated Interview

Applicant thanks Examiner Abdou Seye for his time and consideration of the proposed claim amendments discussed in the telephone interview on October 18, 2007. In the interview Examiner Abdou Seye indicated that it appeared that the claim amendments would overcome the applied art. Examiner Abdou Seye also indicated that further consideration of the applied art and a new search may be conducted upon receiving the response. Examiner Abdou Seye also indicated that further discussion of a restriction of the claims may be necessary upon receiving the response.

Response to Rejections

The disclosure is related to implementing distributed and asynchronous processing in COBOL. As disclosed in paragraphs 006-008 of the disclosure, programming languages such as C and JAVA have functionality for performing distributed and asynchronous processing, such as shared memory and message queues, threads, semaphores and mutexes, events, signal handlers, and sockets. The distributed and asynchronous processing functionality available in C and JAVA was unavailable in COBOL. Because many businesses have legacy applications that have been developed in COBOL that are well suited for performing their intended tasks, it is difficult for businesses to abandon those applications. Some solutions to abandoning the legacy COBOL application is to redeveloping the COBOL applications in C or JAVA or the legacy COBOL applications may be provided with an interface to cooperate with C or JAVA programs. When the legacy COBOL applications are provided with the interface, the C or JAVA programs may then perform the distributed and asynchronous processing tasks that may be necessary in modern business environments.

Rather than redeveloping the COBOL applications or utilizing an interface with another programming language, such as C or JAVA, the disclosure enables COBOL applications to perform distributed and asynchronous processing tasks through a technical layer using functionality native to COBOL. Paragraphs 027-039 of the disclosure provide a detailed description of the technical layer. For example, the technical layer may be COBOL modules, routines, or paragraphs that may be defined within a COBOL library. The COBOL library may then be linked into a COBOL program, such that the COBOL program may utilize the functionality of the technical layer through calls to the COBOL modules, routines, or paragraphs. The claims are directed to the distributed and asynchronous functionality enabled in a COBOL program through the disclosed technical layer.

Miller is directed to generating a program with routines written using code segments that are computer language diverse and/or programming paradigm diverse. Miller discloses in column 7, lines 11-55 to enable communication between the diverse routines event tokens may be used. Miller discloses in column 7, lines 42-45, "All application programs 126 and the functions which make up the application programs 126 are required to conform to the syntax associated with event tokens." Therefore, Miller enables communication between diverse programming routines by requiring the programming routines to conform to a particular syntax when communicating. Therefore, the disclosure of Miller may use a COBOL routine for performing COBOL functions and use a C routine for performing POSIX functions or other non-native COBOL functions similar to other prior art approaches described above.

Obin is directed to converting a procedural program, such as a COBOL program, into an object class in an object oriented programming language. The object class may include a main method corresponding to the main routine of the procedural program and additional methods corresponding to each subroutine in the procedural program. Therefore, a COBOL program is simply converted into an object oriented programming language, such as JAVA similar to other prior art approaches described above.

In light of the differences between the goals of the disclosure and the cited art, the differences between the claim limitations and the cited portions of the prior art are discussed in detail below.

Response to Rejections under Section 103

In the Office Action dated July 26, 2007, Claims 1 - 32 and 47 – 53 were rejected under 35 U.S.C. as being unpatentable over Miller et al., U.S. Patent No. 5,754,855 (hereinafter "Miller") in view of Obin et al., U.S. Patent No. 6,526,569 (hereinafter "Obin").

Claim 1:

I. Miller in view of Obin does not teach or suggest a process identifier associated with the child process.

The Office Action relied on the event token illustrated in Fig. 4A and Fig. 4B of Miller to read on the claimed "index including a process identifier and an event associated with a child process." Applicant respectfully submits that while the event token does include an event identifier 404, the event token does not include a process identifier. Rather, the event token includes a qualifying data pointer 406. The qualifying data pointer 406 points to an address in memory that contains information pertaining to the specific occurrence of the event represented by the event token 402 (Miller: column 8, lines 30-41). For example, Miller discloses that the qualifying data pointer 406 may point to information in a block of memory such as the instruction that was being processed when the event was detected. Applicant respectfully submits that the qualifying data pointer 406 is not a process identifier associated with the child process.

MPEP 2173.05(a)(III) states:

> "In applying the prior art, the claims should be construed to encompass all definitions that are consistent with applicant's use of the term. See *Tex. Digital Sys., Inc. v. Telegenix, Inc.*, 308 F.3d 1193, 1202, 64 USPQ2d 1812, 1818 (Fed. Cir. 2002). It is appropriate to compare the meaning of terms given in technical dictionaries in order to ascertain the accepted meaning of a term in the art. In re Barr, 444 F.2d 588, 170 USPQ 330 (CCPA 1971). >See also MPEP § 2111.01.<"

Applicant respectfully submits that the term "process identifier" is a term of art. The online technical dictionary TechWeb found at the URL http://www.techweb.com/encyclopedia/ defines the term process identifier (PID) as, "A temporary number assigned by the operating system to a process or service." The qualifying data pointer 406 simply identifies an address in memory and does not identify any processes. Further, assuming arguendo that the qualifying data pointer 406 is a process identifier, the qualifying data pointer 406 is not a process identifier of a child process. Rather, the qualifying data pointer 406 may identify the instruction that caused the event token to be generated in the first place.

Applicant notes that the Office Action also cited disclosure of Miller directed to the registration event manager 604 in column 11, lines 55-57. Applicant notes that the registration event manager 604 is part of the event manager 124 and is not maintained "in a COBOL program" as required by Claim 1. Further, the cited portion of Miller does not provide any teaching or suggestion that the registration event manager 604 maintains an index including a process identifier and an event associated with a child process.

II. Miller in view of Obin does not teach or suggest initializing, by the COBOL program, the child process.

Claim 1 has been amended to recite, "initializing, by the COBOL program, the child process". Applicant respectfully submits that no new matter is introduced by this amendment and that support may be found in the specification as originally filed in at least paragraph 092. Applicant respectfully submits that neither Miller nor Obin disclose initializing the child process, much less that the child process is initialized by the COBOL program. A search for the term "initialize" in Miller and Obin did not produce any results. A search for the term "initiate" in Miller and Obin only produced results in column 6, lines 37-48 of Miller, where initiating a transfer of control is discussed. Applicant respectfully submits that this disclosure in Miller does not provide any teaching or suggesting of initializing the child process by the COBOL program.

III. Miller in view of Obin does not teach or suggest placing the child process in a wait state when the child process is initialized.

The Office Action relied on the disclosure of Miller in column 6, lines 49-62 to read on placing the child process in a wait state. The cited portion of Miller discloses synchronous condition events. Miller discloses, "A synchronous signal is an indication of the occurrence of an event whose origin can be attributed to a ... originating thread ... wherein the originating thread and the receiving thread ... execute synchronously." Miller further discloses, "The execution of the originating thread is suspended while the receiving thread is processing the signal." Applicant notes that the originating thread is suspended, not the receiving thread.

Further, Claim 1 has been amended to require that the child process is placed in the wait state "when the child process is initialized". Applicant respectfully submits that no new matter is introduced by this amendment and that support may be found in the specification as originally filed in at least paragraph 092. Applicant notes that the originating thread is suspended after it has been executing and a synchronous event occurs, not after it has been initialized.

IV. Miller in view of Obin does not teach or suggest signaling, by the COBOL program, the child process to run.

Miller discloses that upon an event token being generated by the occurrence of an event, the event manger 124 processes the event token (Miller: column 8, lines 42-54). Therefore, the COBOL program doesn't signal the receiving thread to run. Rather, the event manager 124 would signal the receiving thread to run.

V. Miller in view of Obin does not teach or suggest the event token is maintained by the COBOL program.

The Office Action relied on the event token illustrated in Fig. 4A and Fig. 4B of Miller to read on the claimed index. Assuming arguendo that the event token is the claimed index, Applicant respectfully submits that the event token is not maintained by the COBOL program. Rather, Miller discloses in column 8, lines 42-44, "When a routine detects an event, the routine generates an event token by calling an 'event token generating function' provided by the run-time environment 122. Therefore, the routine doesn't **maintain** the event token as claimed, but rather **generates** the event token through a call to the run-time environment. The routine further **transfers** the generated event token to the event manager 124 for processing (Miller: column 8, lines 42-54).

VI. Miller in view of Obin does not teach or suggest a child process.

Miller discloses that a program may include one or more processes, where each process may include one or more threads, and each thread may include one or more routines (Miller: column 5, lines 15-25). Miller further discloses that a program 126A includes a thread with a COBOL routine 128 that may invoke a PL/I routine 130 (Miller: column 4, lines 8-11 and column 5, lines 47-51). Therefore, the COBOL routine may invoke another routine. Miller does not teach or suggest a child process where a COBOL program signals the child process to run.

VII. Obin does not cure the deficiencies of Miller.

Applicant respectfully submits that Obin does not cure the deficiencies of Miller discussed in sections I-V above. As noted above, Obin is directed to converting a procedural program such as COBOL into an object oriented program. It is unclear how the teachings of Miller and Obin are being combined. Would each of the routines, including the COBOL routine of Miller, be converted into methods of an object class? How would this combination result in the claim limitations? Why would the invention of Miller require or want an object oriented compiler?

For at least the reasons established above in sections I-VII, Applicant respectfully submits that independent Claim 1 is not taught or suggested by Miller in view of Obin and respectfully requests allowance of this claim.

Dependent Claims 2-16 and 19 depend directly or indirectly from independent Claim 1 and incorporate all of the limitations thereof. Accordingly, for at least the reasons established in sections I-VII above, Applicant respectfully submits that Claims 2-16 and 19 are not taught or suggested by Miller in view of Obin and respectfully requests allowance of these claims.

Claim 20:

VIII. Miller in view of Obin does not teach or suggest a second COBOL program.

As noted above, Miller discloses a program 126A includes a thread with a COBOL routine 128 that may invoke a PL/I routine 130 (Miller: column 4, lines 8-11 and column 5, lines 47-51). Miller does not provide any teaching or suggestion of a second COBOL program as required by Claim 20. Applicant respectfully submits that Obin does not cure the deficiencies of Miller.

IX. Miller in view of Obin does not teach or suggest a module maintaining a state sharable between the first and second COBOL programs.

Applicant respectfully submits that Miller in view of Obin does not teach or suggest a module that maintains a state sharable between the first and second COBOL programs to coordinate the processing of the first and second routines of first and second COBOL programs.

For at least the reasons established above in sections VII-IX, Applicant respectfully submits that independent Claim 20 is not taught or suggested by Miller in view of Obin and respectfully requests allowance of this claim.

Dependent Claims 21-28 depend directly or indirectly from independent Claim 20 and incorporate all of the limitations thereof. Accordingly, for at least the reasons established in sections VII-IX above, Applicant respectfully submits that Claims 21-28 are not taught or suggested by Miller in view of Obin and respectfully requests allowance of these claims.

Claim 47:

X. Miller in view of Obin does not teach or suggest registering, by a COBOL language program, a signal handler with an operating system.

While Miller discloses in column 11, lines 53-59 that event handlers for stack frames may be registered with an event manager 124 through a registration event manager 604, Miller does not disclose that the event handlers are registered by a COBOL program or that the event handlers are registered with an operating system as claimed.

For at least the reasons established above in sections VII and X, Applicant respectfully submits that independent Claim 47 is not taught or suggested by Miller in view of Obin and respectfully requests allowance of this claim.

Dependent Claims 48-52 depend directly or indirectly from independent Claim 47 and incorporate all of the limitations thereof. Accordingly, for at least the reasons established in sections VII and X above, Applicant respectfully submits that Claims 48-52 are not taught or suggested by Miller in view of Obin and respectfully requests allowance of these claims.

In the Office Action dated July 26, 2007, Claims 17, 18, 29-32, and 53 were rejected under 35 USC § 103(a) as being unpatentable over Miller in view of Obin and further in view of Keith Haviland, Dina Gray and Ben Salama, Pearson Education Limited, 1998 "Unix System Programming", second edition (hereinafter "Keith").

Claims Depending from Claim 1:

Dependent Claims 17 and 18 depend directly or indirectly from independent Claim 1 and incorporate all of the limitations thereof. Accordingly, for at least the reasons established in sections I-VII above, Applicant respectfully submits that Claims 17 and 18 are not taught or suggested by Miller in view of Obin and further in view of Keith and respectfully requests allowance of these claims. Applicant respectfully submits that Keith does not cure the deficiencies of Miller in view of Obin.

Claims Depending from Claim 20:

Dependent Claims 29-32 depend directly or indirectly from independent Claim 20 and incorporate all of the limitations thereof. Accordingly, for at least the reasons established in sections VII-IX above, Applicant respectfully submits that Claims 21-32 are not taught or suggested by Miller in view of Obin and further in view of Keith and respectfully requests allowance of these claims. Applicant respectfully submits that Keith does not cure the deficiencies of Miller in view of Obin.

Claims Depending from Claim 47:

Dependent Claim 53 depends directly or indirectly from independent Claim 47 and incorporates all of the limitations thereof. Accordingly, for at least the reasons established in sections VII and X above, Applicant respectfully submits that Claim 53 is not taught or suggested by Miller in view of Obin and further in view of Keith and respectfully requests allowance of this claim. Applicant respectfully submits that Keith does not cure the deficiencies of Miller in view of Obin.

New Claims 55-69:

Claims 55-69 are added by this amendment and are respectfully submitted not to add any new matter. Applicant respectfully submits that these claims do not contain any new matter. Support for these claims is found throughout the original disclosure, including paragraphs 090-099.

Claim 55:

Claim 55 includes limitations substantially similar to the limitations discussed in sections I-VII above. For at least the reasons established above in sections I-VII, Applicant respectfully submits that independent Claim 55 is not taught or suggested by Miller in view of Obin and respectfully request allowance of this claim.

Dependent Claims 56-69 depend directly or indirectly from independent Claim 55 and incorporate all of the limitations thereof. Accordingly, for at least the reasons established in sections I-VII above, Applicant respectfully submits that Claims 56-69 are not taught or suggested by Miller in view of Obin and respectfully requests allowance of these claims.

Conclusion

Applicant respectfully submits that the present application is in condition for allowance for the reasons stated above. If the Examiner has any questions or comments or otherwise feels it would be helpful in expediting the application, he is encouraged to telephone the undersigned at (972) 731-2288.

The Commissioner is hereby authorized to charge payment of any further fees associated with any of the foregoing papers submitted herewith, or to credit any overpayment thereof, to Deposit Account No. 21-0765, Sprint.

Respectfully submitted,

[illegible]

Date: October 25, 2007

Michael W. Piper
Reg. No. 39,800

ATTORNEY FOR APPLICANT

CONLEY ROSE, P.C.
5601 Granite Parkway, Suite 750
Plano, Texas 75024
(972) 731-2288
(972) 731-2289 (facsimile)